2012

Space-efficient algorithms for reachability in surface-embedded graphs

Derrick Stolee
University of Nebraska–Lincoln, dstolee@cse.unl.edu

N. V. Vinodchandran
University of Nebraska-Lincoln, vinod@cse.unl.edu

2012 IEEE 27th Conference on Computational Complexity

Space-efficient algorithms for reachability in surface-embedded graphs

Derrick Stolee
Department of Computer Science and Engineering
Department of Mathematics
University of Nebraska–Lincoln
Lincoln, NE, USA
dstolee@cse.unl.edu

N. V. Vinodchandran
Department of Computer Science and Engineering
University of Nebraska–Lincoln
Lincoln, NE, USA
vinod@cse.unl.edu

***Abstract*—This work presents a log-space reduction which compresses an n-vertex directed acyclic graph with $m(n)$ sources embedded on a surface of genus $g(n)$, to a graph with $O(m(n) + g(n))$ vertices while preserving reachability between a given pair of vertices. Applying existing algorithms to this reduced graph yields new *deterministic* algorithms with improved space bounds as well as improved simultaneous time-space bounds for the reachability problem over a large class of directed acyclic graphs. Specifically, it significantly extends the class of surface-embedded graphs with log-space reachability algorithms: from planar graphs with $O(\log n)$ sources, to graphs with $2^{O(\sqrt{\log n})}$ sources embedded on a surface of genus $2^{O(\sqrt{\log n})}$. Additionally, it yields an $O(n^{1-\epsilon})$ space algorithm with polynomial running time for reachability over graphs with $O(n^{1-\epsilon})$ sources embedded on surfaces of genus $O(n^{1-\epsilon})$.**

***Keywords*-reachability; surface-embedded graphs; acyclic digraph; log-space algorithm;**

I. Introduction

Graph reachability problems are central to space-bounded computations. Different versions of this problem characterize several important space complexity classes. The problem of deciding whether there is a path from a given vertex u to a vertex v in a directed acyclic graph is the canonical complete problem for non-deterministic log-space (NL). The recent breakthrough result of Reingold implies that the undirected reachability problem characterizes the complexity of deterministic log-space (L) [1]. It is also known that certain restricted promise versions of the directed reachability problem characterize randomized log-space computations (RL) [2]. Clearly, progress in space complexity studies is directly related to progress in understanding graph reachability problems. We refer the readers to a (two decades old, but excellent) survey by Avi Wigderson [3] and a recent update by Eric Allender [4] to further understand the significance of reachability problems in complexity theory.

In this paper we focus on designing *deterministic* algorithms for reachability with improved space complexity. For the general directed graph reachability problem the best known result remains the 40-year old $O(\log^2 n)$ space bound due to Savitch [5] (where n is the number of vertices in the graph). Designing a deterministic algorithm for the directed graph reachability problem that asymptotically beats Savitch's bound is the most significant open questions in this topic. While this remains a difficult open problem, investigating classes of directed graphs for which we can design space efficient algorithms that beat Savitch's bound is an important research direction with some outstanding results, including Saks and Zhou's $O(\log^{3/2} n)$ bound for reachability problems characterizing RL computations [6] and Reingold's log-space algorithm for the undirected reachability problem [1]. In this paper we consider the reachability problem over *directed acyclic graphs that are embedded on topological surfaces*. We present the best (to date) space complexity upper bounds for the reachability problem over this class of directed graphs.

Prior Results: Jakoby, Liśkiewicz, and Reischuk [7] and Jakoby and Tantau [8] show that various reachability and optimization questions for *series-parallel* graphs admit deterministic log-space algorithms. Series-parallel graphs are a very restricted subclass of planar DAGs. In particular, such graphs have a single source and a single sink. Allender, Barrington, Chakraborty, Datta, and Roy [9] extended the result of Jakoby *et al.* to show that the reachability problem for Single-source Multiple-sink Planar DAGs (SMPDs) can be decided in logarithmic space. Building on the work of Allender *et al.* [9], in [10], the present authors show that reachability for planar DAGs with $O(\log n)$ sources can be decided in logarithmic space. Theorem 1 below is implicit in [10].

Theorem 1 ([10])**.** *Let $\mathcal{G}(m)$ denote the class of planar DAGs with at most $m = m(n)$ sources, where n is the number of vertices. The reachability problem over $\mathcal{G}(m)$ can be solved by a log-space nondeterministic machine using a one-way certificate of $O(m)$ bits. In particular, reachability over $\mathcal{G}(m)$ can be decided deterministically in $\min\{O(\log n + m), O(\log n \cdot \log m)\}$ space.*

The $O(\log n + m)$ space bound is obtained by a brute-force search over all certificates of length $O(m)$. Setting $m = O(\log n)$ we get a deterministic log-space algorithm for reachability over planar graphs with $O(\log n)$ source nodes. The $O(\log n \cdot \log m)$ bound is obtained by first

1093-0159/12 $26.00 © 2012 IEEE
DOI 10.1109/CCC.2012.15

converting the nondeterministic algorithm to a layered graph with m layers and $\mathrm{poly}(n)$ vertices in each layer, and then applying Savitch's algorithm on this layered graph. The second bound leads to a deterministic algorithm that beats Savitch's bound for reachability over DAGs with $2^{o(\log n)}$ sources (for example, setting $m = 2^{\log^{1-\epsilon} n}$, it gives a $\log^{2-\epsilon} n$ space algorithm for reachability over planar graphs with $2^{\log^{1-\epsilon} n}$ source nodes).

However, if we are aiming for deterministic algorithms with $O(\log n)$ space complexity, the above theorem could not handle asymptotically more than $\log n$ sources. In this paper we improve the upper bound from $\min\{O(\log n + m), O(\log n \cdot \log m)\}$ to $O(\log n + \log^2 m)$. This yields a new deterministic log-space algorithm for reachability over planar DAGs with $m = 2^{O(\sqrt{\log n})}$ source nodes. We also extend our results to graphs embedded on higher genus surfaces. In addition, techniques of this paper also lead to new results on simultaneous time-space bounds for reachability which are not implied by [10].

The main technique of [10] (that leads to $O(\log n \cdot \log m)$ bound) can be viewed as a log-space reduction that takes $\langle G, u, v\rangle$ where $G \in \mathcal{G}(m)$ and outputs $\langle G', u', v', \rangle$ so that (a) there is a directed path from u to v in G if and only if there is a directed path from u' to v' in G', (b) G' is a layered graph with m layers and $\mathsf{poly}(n)$ vertices per layer. This $\mathsf{poly}(n)$ factor in the size of G' makes it useless for obtaining a logarithmic space bound. We get rid of this poly(n) factor by avoiding the intermediate nondeterminism and giving a direct reduction to a new reachability instance. This requires a more careful analysis of the topological interaction of paths in surface-embedded graphs.

New Results. Let n be the number of vertices in the input graph. Let $\mathcal{G}(m, g)$ denote the class of DAGs with at most $m = m(n)$ source vertices embedded on a surface (orientable or non-orientable) of genus at most $g = g(n)$. Our main technical contribution is the following log-space reduction that *compresses* an instance of reachability for such surface-embedded DAGs.

Theorem 2. *There is a log-space reduction that given an instance $\langle G, u, v\rangle$ where $G \in \mathcal{G}(m, g)$ and u, v vertices of G, outputs an instance $\langle G', u', v'\rangle$ where G' is a directed graph and u', v' vertices of G', so that (a) there is a directed path from u to v in G if and only if there is a directed path from u' to v' in G', (b) G' has $O(m + g)$ vertices.*

By a direct application of Savitch's theorem on the reduced instance we get the following result.

Theorem 3. *The reachability problem for graphs in $\mathcal{G}(m, g)$ can be decided in deterministic $O(\log n + \log^2(m + g))$ space.*

This improves the earlier-known space bound of $\min\{O(\log n + m), O(\log n \cdot \log m)\}$ and also extends it to higher genus graphs.

By setting $m = g = 2^{O(\sqrt{\log n})}$ we get a deterministic log-space algorithm for reachability over graphs in $\mathcal{G}(2^{O(\sqrt{\log n})}, 2^{O(\sqrt{\log n})})$.

Corollary 4. *The reachability problem for directed acyclic graphs with $2^{O(\sqrt{\log n})}$ sources embedded on surfaces of genus $2^{O(\sqrt{\log n})}$ can be decided in deterministic logarithmic space.*

By setting m and g to be $n^{o(1)}$ we get $o(\log^2 n)$ bound. The following corollary as stated is implicit in [10]. However, the space bound we get for any specific function $n^{l(n)}$ where $l(n) \in o(1)$ is better than what is implied by the results of [10].

Corollary 5. *The reachability problem for directed acyclic graphs embedded on surfaces with* sub-polynomial genus *and with* sub-polynomial number of sources *can be decided in deterministic space $o(\log^2 n)$.*

Theorem 2 leads to new simultaneous time-space bound for the reachability problem. Designing algorithms for reachability with simultaneous time and space bound is another important direction that has been of considerable interest in the past. Since a depth first search can be implemented in linear time and linear space, the goal here is to improve the space bound while maintaining a polynomial running time. The most significant result here is Nisan's $O(\log^2 n)$ space, $n^{O(1)}$ time bound for RL [11]. The best upper bound for general directed reachability is the 20-year old $O(n/2^{\sqrt{\log n}})$ space, $n^{O(1)}$ time algorithm due to Barnes, Buss, Ruzzo and Schieber [12]. Combining our reduction with a simple depth-first search gives better simultaneous time-space bound for reachability over a large class of graphs that beats the Barnes *et al.* bound.

Theorem 6. *The reachability problem for graphs in $\mathcal{G}(m, g)$ can be decided in polynomial time using $O(\log n + m + g)$ space.*

Note that Theorem 6 has a space bound which matches the $O(\log n + m)$ space bound of Theorem 1, except it guarantees polynomial time, where the previous bound gave $2^{O(m)}\,\mathrm{poly}(n)$ running time. For any $\epsilon < 1$, we get a polynomial time algorithm for reachability over graphs in $\mathcal{G}(O(n^\epsilon), O(n^\epsilon))$ that uses $O(n^\epsilon)$ space.

Corollary 7. *For any ϵ with $0 < \epsilon < 1$, the reachability problem for graphs in $\mathcal{G}(O(n^\epsilon), O(n^\epsilon))$ can be decided in polynomial time using $O(n^\epsilon)$ space.*

We note that the upper bound on space given in Theorem 6 can be slightly improved to $O\left((m + g)2^{-\sqrt{\log(m+g)}}\right)$ by using the Barnes *et al.* algorithm instead of depth-first search, which will give a $o(n^\epsilon)$ space bound in the above corollary.

Theorem 8. *The reachability problem for graphs in $\mathcal{G}(m,g)$ can be decided in deterministic polynomial time using $O\left(\log n + \frac{m+g}{2^{\sqrt{\log(m+g)}}}\right)$ space.*

Before we go into further details, we note that throughout this paper certain known log-space primitives are frequently used as subroutines without explicit reference to them. In particular, Reingold's log-space algorithm for undirected reachability is often used, for example to identify connected components in certain undirected graphs.

II. Preliminaries

We mainly deal with directed graphs. A directed edge $e = xy$ has the direction from x to y and we call x the *tail* denoted by $\mathrm{Tail}(e)$, and y the *head* denoted by $\mathrm{Head}(e)$. We assume that the input graph G is embedded on a surface S where every face is homeomorphic to an open disk. Such embeddings are called *2-cell embeddings*. We assume that such an embedding is presented as a *combinatorial embedding* where for each vertex v the circular ordering of the edges incident to v is specified. In the case of a non-orientable surface, the signature of an edge is also given, specifying if the orientation of the rotation switches across this edge. Since computing or approximating a low-genus embedding of a non-planar graph is an NP-complete problem [13], [14], we require the embedding to be given as part of the input and we consider reachability in $\mathcal{G}(m,g)$ to be a promise problem. In the case of genus zero, we can compute a planar embedding in log-space and the promise condition can be removed.

Let G be a graph with n vertices and e edges embedded on a surface S with f faces. Then by the well known *Euler's Formula* we have $n - e + f = \chi_S$, where χ_S is the Euler characteristic of the surface S. The number of faces in a graph is log-space computable from a combinatorial embedding (for a proof, see [15]), so χ_S is also computable in log-space. The genus g_S of the surface S is given by the equation $\chi_S = 2 - 2g_S$ for orientable surfaces and $\chi_S = 2 - g_S$ for non-orientable surfaces.

Let C be a simple closed curve on S given by a cycle in the underlying undirected graph of G. C is called *surface separating* if the removal of C disconnects G. A surface separating curve C is called *contractible* if removal of the nodes in C disconnects G where at least one of the connected components is homeomorphic to a disc. Given a cycle C it is possible to detect the type of C in log-space (for example by using the log-space algorithm for undirected reachability to find the connected components, then calculating the Euler characteristic for each component).

We assume that the given graph is acyclic. Lemma 1.9 in the appendix gives a technique for converting a source-bounded reachability algorithm on graphs promised to be acyclic into a cycle-detection algorithm without asymptotically increasing the space requirement.

A. Forest Decomposition, Edge Classification, and Topological Equivalence

A simple structural decomposition, called a *forest decomposition*, of a directed acyclic graph forms the basis of our algorithm. This forest decomposition has been utilized in previous works [9], [10].

Let G be a directed acyclic graph and let u, v be two vertices. Our goal is to decide whether there is a directed path from u to v. Let $u, s_1, \ldots, s_m$ be the sources of G. If u is not a source, make it a source by removing all the incoming edges. This does not affect uv-reachability, increases the number of sources by at most one, and only reduces the genus of the embedding.

Let A be a deterministic log-space algorithm that on input of a non-source vertex x, outputs an incoming edge yx (for example, selecting the lexicographically-first vertex y so that yx is an edge in G). This algorithm defines a set of edges $F_A = \{yx : x \in V(G) \setminus \{u, v, s_1, \ldots, s_m\},\ y = A(x)\}$, called a *forest decomposition* of G.

Since G is acyclic, the reverse walk $x_1, x_2, \ldots,$ where $x_1 = x$ and $x_{i+1} = A(x_i)$, must terminate at a source s_j, u, or v, so the edges in F_A form a forest subgraph. For the purposes of the forest decomposition, v is treated as a source since no incoming edge is selected. If a vertex x is in the tree with source v, then all non-tree edges entering x are deleted. This does not affect uv-reachability, since G is acyclic and does not increase the number of sources or the genus of the surface. Each connected component in F_A is a tree rooted at a source vertex, called a *source tree*. The forest forms a typical *ancestor* and *descendant* relationship within each tree. For the remainder of this work, we fix an acyclic graph $G \in \mathcal{G}(m,g)$ embedded on a surface S (defined by the combinatorial embedding) and $F = F_A$ a log-space computable forest decomposition.

Let x and y be two vertices in some source tree T of F. The *tree curve* at xy is the curve on S formed by the unique undirected path in T from x to y. If xy is an edge, then the closed curve formed by xy and the tree curve at xy is called the *closed tree curve* at xy. Edges in G that are not included in F can be partitioned into two classes, *local* and *global*. We use this classification to create subgraphs of G which are locally embedded on a disk.

Given an S-embedded graph G and a forest decomposition F, an edge xy in $E(G) \setminus F$ is classified as *local*[1] if (a) x and y are on the same tree in F, (b) the closed tree curve at xy is contractible (i.e. the curve cuts S into a disk and another surface), and (c) No sources lie on the interior of the surface which is homeomorphic to a disk. If S is the sphere, then the curve cuts S into two disks and xy is local if one of the disks contains no source in the interior. Otherwise, xy is *global*.

[1]This definition of *local* differs from the use in [9] and [10].

Let T be a connected component in the forest decomposition F along with the local edges between vertices in T. The *region* of T, denoted $\mathcal{R}[T]$ is the portion of the surface S given by the faces enclosed by the tree and local edges in T. The faces that compose $\mathcal{R}[T]$ are together homeomorphic to a disk, since $\mathcal{R}[T]$ can contract to the source vertex by contracting the disks given by the local edges into the tree, and then contracting the tree into the source vertex. This disk is oriented using the combinatorial embedding at the source by the right-hand rule. Reachability in such subgraphs T can be decided using the SMPD algorithm [9], in log-space. Note that the restriction of a 2-cell embedding implies all global edges are incident to vertices on the outer curve of the disk $\mathcal{R}[T]$. Our figures depict source trees as circles, with the source placed in the center, with tree edges spanning radially away from the source[2]. We can also assign a clockwise or counter-clockwise direction to all local edges in a source tree region $\mathcal{R}[T]$. For a local edge xy, the closed tree curve at xy is cyclicly oriented by the direction of xy. The edge xy is considered clockwise (counter-clockwise) if this cyclic orientation is clockwise (counter-clockwise) with respect to the orientation of $\mathcal{R}[T]$.

The following notion of topological equivalence plays a central role in our algorithms. It was originally presented in [10] for planar graphs, but we extend it to arbitrary surfaces.

Let G be a graph embedded on a surface S. Let F be a forest decomposition of G. We say two (undirected) global edges xy and wz are *topologically equivalent* if the following two conditions are satisfied: (a) They span the same source trees in F (assume x and w are on the same tree), (b) The closed curve in the underlying undirected graph formed by (1) the edge xy, (2) the tree curve from y to z, (3) the edge zw, and (4) the tree curve from w to x bounds a connected portion of S, denoted $D(xy, wz)$, that is homeomorphic to a disk and no source lies within $D(xy, wz)$.

In fact, topological equivalence is an equivalence relation. Let E be an equivalence class of global edges containing an edge e, where e spans two different source trees. Consider the subgraph of G given by the vertices in the source trees containing the endpoints of e, along with all local edges in those trees and the edges in E. This subgraph is embedded in a disk on S (for a proof, see Corollary 4.3 in the appendix). We shall make explicit use of this locally-planar embedding. For an equivalence class of global edges spanning vertices in the same tree, a similar subgraph and embedding is formed by considering the ends of the equivalence class to be different copies of that source tree.

The lexicographically-least edge e in a topological equivalence class of global edges is log-space computable. By counting how many global edges which are lexicographically smaller than e and are the lexicographically-least in their equivalence classes, the equivalence class containing e is assigned an index i. The class E_i is the ith equivalence class in this ordering. We shall use this notation to label the equivalence classes.

Let E_i be an equivalence class of global edges. Define the *region enclosed by* E_i as $\mathcal{R}[E_i] = \bigcup_{e_1,e_2 \in E_i} D(e_1, e_2)$. The region $\mathcal{R}[E_i]$ has some properties which are quickly identified. There are two edges $e_a, e_b \in E_i$ so that $\mathcal{R}[E_i] = D(e_a, e_b)$. These outer edges define the *sides* of $\mathcal{R}[E_i]$. The *boundary* of $\mathcal{R}[E_i]$ is given by these two edges and their ancestor paths in F on all four endpoints. All vertices in a source tree T are contained in the region $\mathcal{R}[T]$. Let T_A and T_B be the two source trees containing the tail and head, respectively, of the representative edge in E_i. The vertices within the boundary of $\mathcal{R}[E_i]$ are within $\mathcal{R}[T_A]$ and $\mathcal{R}[T_B]$. The vertices in $\mathcal{R}[E_i]$ are partitioned into two *ends*, A and B, where the vertices are placed in an end determined by containment in $\mathcal{R}[T_A] \cap \mathcal{R}[E_i]$ and $\mathcal{R}[T_B] \cap \mathcal{R}[E_i]$ when the trees T_A and T_B are different or by the two connected components of $\mathcal{R}[T_A] \cap \mathcal{R}[E_i]$ when the trees are equal. Note that the endpoints of edges in E_i lie on the boundary of the regions $\mathcal{R}[T_A]$ and $\mathcal{R}[T_B]$. There is an ordering $e_a = e_1, e_2, \ldots, e_k = e_b$ of E_i so that the endpoints of the e_j on the A-end appear in a clockwise order in that tree. Two regions $\mathcal{R}[E_i]$ and $\mathcal{R}[E_j]$ on different classes E_i and E_j intersect only on the boundary paths. The vertices on the boundary are not considered *inside* the region, since they may be in multiple regions.

Since global edges appear on the boundary of $\mathcal{R}[T]$ for a given source tree T, there is a natural clockwise ordering on these edges, with respect to the orientation of T. Further, we can order the incident equivalence classes (with possibly a single repetition, in the case of global edges with both endpoints in T) by the clockwise order the ends $\mathcal{R}[E_i] \cap \mathcal{R}[T]$ appear on the boundary of $\mathcal{R}[T]$.

The resource bounds we prove directly depends on the number of equivalence classes. The following lemma bounds the number of equivalence classes.

Lemma 26. *Let G be a graph embedded on a surface S with Euler characteristic χ_S with a forest decomposition F with m sources. There are at most $3(m + |\chi_S|)$ topological equivalence classes of global edges. If g_S is the genus of S, $|\chi_S| = O(g_S)$ and there are $O(m + g_S)$ equivalence classes of global edges.*

At this point, we take a very different approach than [10]. The algorithm described in [10] focused on reachability within the regions $\mathcal{R}[T]$ on the source trees T. Here, we focus on reachability within and between equivalence classes E_i. We create a constant number of vertices derived from each equivalence class. This constant is given by the number of distinct ways a path can enter the region $\mathcal{R}[E_i]$, use edges in E_i, then leave the region $\mathcal{R}[E_i]$. We call these *patterns*.

[2]This visualization of source trees was crucial to the development of this work, and is due to [9].

III. Patterns in Equivalence Classes

If a directed path P from u to v exists in G, then P leaves the tree rooted at u and travels through the other source trees and global edge equivalence classes before taking a global edge to v. The crucial observation to this work is the following: If we focus on certain "nice" paths, then there are a finite number of ways such paths can enter and exit the region of an equivalence class. Thus for each equivalence class E_i, there are a finite number of "patterns" we need to consider. In this section we formalize these notions. First we define what we mean by a "nice" path.

Let G be a DAG and F be a forest decomposition of G. Let $P = x_1, \ldots, x_k$ be a directed path in G. P is said to be *irreducible* if whenever a vertex x_i appears before x_j in P and x_j is a descendant of x_i in some tree of F, then P follows the edges in F from x_i to x_j. Note that if a path exists between two vertices, an irreducible path also exists, by swapping violating subpaths with the appropriate tree paths. For an irreducible path between u and v, the portions of this path within each source tree must follow a clockwise or counter-clockwise direction on each local edge (see Lemma 17 in the appendix for an exact statement). We associate *right* and *left* with the rotational directions clockwise and counter-clockwise, respectively.

Consider an equivalence class E_i between source trees T_A and T_B, a rotational direction d (clockwise or counterclockwise), and a vertex x in T_A outside the region $\mathcal{R}[E_i]$. We say that the vertex x *fully reaches* E_i in the direction d if there is an irreducible d-directional local path from x to an endpoint of each edge in E_i. If x does not fully reach E_i in direction d, but there is a local path from x to an endpoint of some edge of E_i, then we say x *partially reaches* E_i in this direction. If such a path is irreducible, then the path follows a clockwise or counter-clockwise direction within T_A and we say x fully (or partially) reaches E_i *using a clockwise (or counter-clockwise) rotation*.

Lemma 27. *Let x be a vertex in a source tree T_A. For each rotational direction d (clockwise or counter-clockwise), there is an ordering $E_{i_0}, E_{i_1}, \ldots, E_{i_\ell}$ of the edge classes reachable via irreducible d-directional paths so that x fully reaches each E_{i_j} in direction d for $j \in \{1, \ldots, \ell - 1\}$, x either fully or partially reaches E_{i_0} and E_{i_ℓ} in direction d, and if x is not in the interior of $\mathcal{R}[E_{i_0}]$, x fully reaches E_{i_0}.*

Each irreducible path P between two vertices x and y induces a list of edge classes $E_{i_1}, \ldots, E_{i_\ell}$ for some ℓ where the global edges of P visit each class E_{i_j} in order of increasing j, and $E_{i_j} \neq E_{i_{j+1}}$ for each $j \in \{1, \ldots, \ell - 1\}$. This list is the *induced class list* for the path P. Once the induced class list is known, the path P must take local paths between edges in E_{i_j} and edges in $E_{i_{j+1}}$. Since P is irreducible, these local paths have a clockwise or counterclockwise direction, determined by the orientation of the path (which may agree or disagree with the orientation of the current tree). While this local path traverses from E_{i_j} to $E_{i_{j+1}}$, it must cross the boundary of $\mathcal{R}[E_{i_j}]$ and the boundary of $\mathcal{R}[E_{i_{j+1}}]$ at an ancestor path of a boundary edge. There are only two possible ends (A or B) where these local paths can start and end, only two possible rotational directions (R and L for *right* and *left*), and two possible orientations ($+$ or $-$ with respect to the current tree). Note that if P crosses a boundary path to enter $\mathcal{R}[E_{i_j}]$, crossing that boundary again would either create a cycle or would violate irreducibility. Hence, an irreducible path must cross two boundaries of $\mathcal{R}[E_{i_j}]$.

Specifically, the entrance direction, the exit end, and the exit direction combine into a *pattern* which has the most important information about the behavior of P within E_i.

Definition 28 (The Pattern Set)**.** Let E_i be an equivalence class of global edges. An irreducible path P that involves an edge of the class E_i *induces* a pattern on E_i defined by $\langle abc \rangle$ with $a, c \in \{\mathrm{L}, \mathrm{R}\}$, $b \in \{\mathrm{S}, \mathrm{X}\}$ where a is the clockwise (R) or counter-clockwise (L) direction the path takes as it enters $\mathcal{R}[E_i]$, c is the direction the path takes as it leaves $\mathcal{R}[E_i]$, and if $b = \mathrm{S}$, the path enters and leaves $\mathcal{R}[E_i]$ on the same end and if $b = \mathrm{X}$, the path enters and leaves $\mathcal{R}[E_i]$ on opposite ends. Define the *pattern set*, $\mathcal{P} = \{\langle \mathrm{RSR} \rangle, \langle \mathrm{LSL} \rangle, \langle \mathrm{RXR} \rangle, \langle \mathrm{RXL} \rangle, \langle \mathrm{LXR} \rangle, \langle \mathrm{LXL} \rangle\}$.

The patterns $\langle \mathrm{LSR} \rangle$ and $\langle \mathrm{RSL} \rangle$ are omitted since they are not realizable by an irreducible path. The four patterns $\langle \mathrm{RSR} \rangle, \langle \mathrm{LSL} \rangle, \langle \mathrm{RXL} \rangle$, and $\langle \mathrm{LXR} \rangle$ are called *full* patterns. Patterns $\langle \mathrm{RXR} \rangle$ and $\langle \mathrm{LXL} \rangle$ are called *nesting* patterns. These names refer to specific properties — properties which are investigated in following sections — that are revealed when paths attempt to induce these patterns. The main difference between the full and the nesting patterns is that nesting patterns have the entrance and exit on the same side of the region, while full patterns involve both sides of the region. A path through the region from one side to the other (thus inducing a full pattern) will force all edges in the class to be reachable. See Table I for figures of these patterns.

Let E_i be an edge class and $\mathcal{R}[E_i]$ be the enclosed region. Let t be an end of $\mathcal{R}[E_i]$ (either A or B) and fix an orientation on that end and a pattern p that involves E_i. Then the *entrance* (*exit*) of the pattern at the t-end is the ancestor path on the boundary of $\mathcal{R}[E_i]$ on the t-end that a path must cross *before* (respectively, *after*) using the edges in E_i that induce the pattern p with the given orientation. (See Figure 4 in the appendix for a visual representation of the entrance and exit of a pattern.) We can now define *pattern descriptions* which are the vertices of the pattern graph that we will define in the next section.

Definition 29 (Pattern Descriptions)**.** Let k be the number of topological equivalence classes of edges of G. A *pattern description* is a tuple $\mathbf{x} = (i, t, o, p)$ where $i \in \{1, \ldots, k\}$,

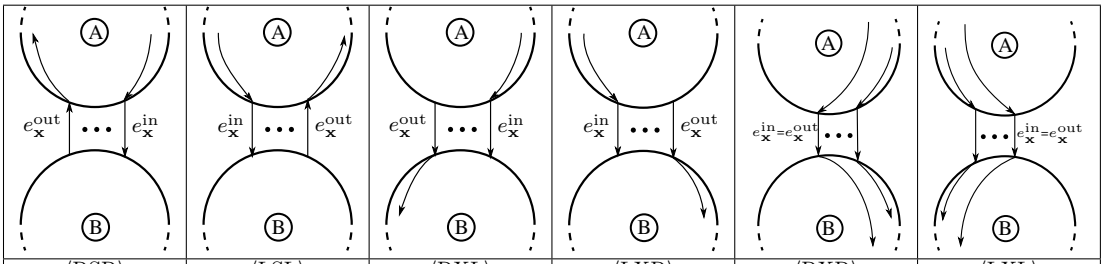

Table I
DIFFERENT PATTERNS USING AN EDGE CLASS E_i, ENTERING FROM THE A-END OF $\mathcal{R}[E_i]$.

$t \in \{A, B\}$, $o \in \{+1, -1\}$, and $p \in \mathcal{P}$. Here i represents the equivalence class E_i, t represents the end of $\mathcal{R}[E_i]$ that contains the entrance, $o \in \{+1, -1\}$ specifies if the orientation of the path is in agreement with (or opposite to, respectively) the local orientation of the tree on the t-side of E_i, and $p \in \mathcal{P}$ represents the pattern used in E_i. The set $\{1, \ldots, k\} \times \{A, B\} \times \{+1, -1\} \times \mathcal{P}$ of all pattern descriptions is denoted by V_{P}.

For example, the description $(i, B, +1, \langle \text{RXL} \rangle)$ is an element in V_{P} corresponding to a $\langle \text{RXL} \rangle$ pattern, using at least one edge of the class E_i starting at the B-side and leaving the A-side, oriented to agree with the B-side. Lemma 26 implies the number of descriptions is $O(m+g_S)$ where m is the number of sources and g the genus of the surface. A pattern description can be represented with $\lceil \log k \rceil + 5 = O(\log(m + g_S))$ bits[3].

We now investigate some properties of paths that induce these pattern descriptions. We focus on a path which uses local edges and global edges in a single equivalence class and induces a single pattern on that class. These single-pattern paths will be concatenated to make larger paths once the structure of the shorter paths is understood.

An important property of these patterns is that if the pattern is of full type or the equivalence class is fully reachable, we can assume without loss of generality that the path used two special edges, which we call the *canonical edge pair*.

Definition 30 (Canonical Edge Pair)**.** Let $\mathbf{x} = (i, t, o, p)$ be a pattern description centered at the edge class E_i. There are two edges (*incoming* and *outgoing*) in E_i, called the *canonical edge pair* for $\mathbf{x}$. The *outgoing edge*, $e^{\text{out}}_{\mathbf{x}}$, is the edge $e \in E_i$ with head on the exit end that is farthest from the exit side so that there exists a local path from $\text{Head}(e)$ to the exit of $\mathcal{R}[E_i]$. The *incoming edge*, $e^{\text{in}}_{\mathbf{x}}$, is the edge $e \in E_i$ with the tail on the entrance end that is closest to the entrance side so that either $e = e^{\text{out}}_{\mathbf{x}}$ or $\text{Tail}(e^{\text{out}}_{\mathbf{x}})$ is reachable from $\text{Head}(e)$ using local paths and edges in E_i.

[3]This bland fact is in fact very important for the later use of Savitch's Theorem.

Full patterns are named so because a path which induces a full pattern intersects the ancestor path of at least one endpoint of every edge in the class. Hence, every edge is reachable. This leads to the property that if an irreducible path induces such a pattern, then the path might as well use the canonical edges in the corresponding equivalence class.

Lemma 31. *Let $\mathbf{x}$ be a pattern description of full type centered at an edge class E_i. Let $y, z \in V(G)$ be vertices not inside $\mathcal{R}[E_i]$, where y is in the source tree on the entrance end of $\mathbf{x}$ and z is in the source tree on the exit end of $\mathbf{x}$. Then there is a path from y to z in G using only local paths and edges of the class E_i that induces the pattern $\mathbf{x}$ if and only if $\text{Tail}(e^{\text{in}}_{\mathbf{x}})$ is reachable from y using a local path in the entrance direction of $\mathbf{x}$ and z is reachable from $\text{Head}(e^{\text{out}}_{\mathbf{x}})$ using a local path in the exit direction of $\mathbf{x}$.*

Lemma 32. *Let $\mathbf{x}$ be a pattern description of full type. The canonical edge pair $(e^{\text{in}}_{\mathbf{x}}, e^{\text{out}}_{\mathbf{x}})$ is log-space computable.*

Nesting patterns are named so because irreducible paths which induce such patterns use exactly one edge of this class, and we may assume that the edge used is the one farthest from the entrance that is reachable (and that a local path exists from its head to the exit). The following lemmas describe properties of nesting patterns.

Lemma 33. *If an irreducible path using local paths and edges in a global edge class E_i induces a nesting pattern, then the path uses exactly one edge in the class E_i.*

Lemma 34. *Let $\mathbf{x}$ be a pattern description of nesting type centered at a global edge class E_i. Then, $e^{\text{in}}_{\mathbf{x}} = e^{\text{out}}_{\mathbf{x}}$, and $e^{\text{out}}_{\mathbf{x}}$ is log-space computable.*

While it would be useful to have a property similar to Lemma 31 for nesting patterns, there may exist a vertex w from which there are paths that induce a nesting pattern without reaching the canonical incoming edge. We can define a new edge in the class that is similarly canonical, except with respect to the vertex w.

Let $\mathbf{x} = (i, t, o, p)$ be a pattern description of nesting type and w be a vertex not in the interior of $\mathcal{R}[E_i]$. The *most-interior* edge of $\mathbf{x}$ reachable from w, denoted $e_{\mathbf{x}}^{\mathrm{int}(w)}$, is the edge e in the class E_i that is farthest from the entrance side of $\mathcal{R}[E_i]$ so that (a) there is a local path from w to $\mathrm{Tail}(e)$ in the entrance direction, and (b) there is a local path from $\mathrm{Head}(e)$ to the exit boundary of $\mathcal{R}[E_i]$.

Lemma 37. *Let $\mathbf{x}$ be a pattern description of nesting type and w a vertex not in the interior of $\mathcal{R}[E_i]$. The most-interior edge, $e_{\mathbf{x}}^{\mathrm{int}(w)}$, is log-space computable. For any vertex z not in $\mathcal{R}[E_i]$, there is a path from w to z that induces the pattern $\mathbf{x}$ if and only if there is an irreducible local path from $\mathrm{Head}\left(e_{\mathbf{x}}^{\mathrm{int}(w)}\right)$ to z in the exit direction of $\mathbf{x}$. If w fully reaches E_i, then $e_{\mathbf{x}}^{\mathrm{int}\,w} = e_{\mathbf{x}}^{\mathrm{out}}$.*

IV. The Pattern Graph

We now describe a graph on $O(m + g_S)$ vertices that preserves uv-reachability.

Definition 38 (The Pattern Graph)**.** Given G and F as above, the *pattern graph*, denoted $\mathrm{P}(G, F) = (V'_{\mathrm{P}}, E'_{\mathrm{P}})$ is a directed graph defined as follows. The vertex set $V'_{\mathrm{P}} = \{u', v'\} \cup V_{\mathrm{P}} = \{u', v'\} \cup (\{1, \ldots, k\} \times \{A, B\} \times \{+1, -1\} \times \mathcal{P})$. For two pattern descriptions $\mathbf{x}, \mathbf{y} \in V_{\mathrm{P}}$, an edge $\mathbf{x} \to \mathbf{y}$ is in E'_{P} if and only if there exists a (possibly empty) list of nesting pattern descriptions $\mathbf{z}_1, \ldots, \mathbf{z}_\ell$ (called an *adjacency certificate*), so that the following two conditions hold:

1) There is an irreducible path from $\mathrm{Head}(e_{\mathbf{x}}^{\mathrm{out}})$ to $\mathrm{Tail}(e_{\mathbf{y}}^{\mathrm{in}})$ which induces the sequence $\mathbf{z}_1, \ldots, \mathbf{z}_\ell$ of nesting pattern descriptions.
2) For each $j \in \{1, \ldots, \ell\}$, $\mathrm{Tail}(e_{\mathbf{z}_j}^{\mathrm{in}})$ is not reachable from $\mathrm{Head}(e_{\mathbf{x}}^{\mathrm{out}})$ using irreducible paths that induce the pattern descriptions $\mathbf{z}_1, \ldots, \mathbf{z}_{j-1}$.

In addition, for a description $\mathbf{x} = (i, t, o, p)$ there is an edge $u' \to \mathbf{x}$ in E'_{P} if and only if $\mathbf{x}$ has the t-end in the tree T_u. Also, for a pattern description $\mathbf{x} = (i, t, o, p)$ there is an edge $\mathbf{x} \to v'$ in E'_{P}, if and only if the class E_i is incident to v, t is the other end of the class, and $p \in \{\langle \mathrm{RXL} \rangle, \langle \mathrm{LXR} \rangle\}$.

Theorem 39. *There is a path from u to v in G if and only if there is a path from u' to v' in $\mathrm{P}(G, F)$.*

Proof: ($\Rightarrow$) Let P be an irreducible path from u to v in G. P induces a sequence of pattern descriptions $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$. Note that $\mathbf{x}_1$ is centered at an edge class that is incident to T_u and the entrance end is on T_u. Note also that $\mathbf{x}_\ell$ is centered at an edge class where the edges have head v. Thus, in $\mathrm{P}(G, F)$, $u' \to \mathbf{x}_1$ and $\mathbf{x}_\ell \to v'$ are edges.

For full pattern descriptions $\mathbf{x}_i$, Lemma 31 implies that we may assume the first edge in the global edge class of $\mathbf{x}_i$ used by P is $e_{\mathbf{x}_i}^{\mathrm{in}}$ and the last such edge is $e_{\mathbf{x}_i}^{\mathrm{out}}$.

Fix $i \in \{1, \ldots, \ell - 1\}$ and let $\mathbf{x}_j$ be the next full pattern induced after $\mathbf{x}_i$. If $j = i + 1$, then the path P takes a local path between the edges that induce the patterns $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$. By Lemma 31, $e_{\mathbf{x}_j}^{\mathrm{in}}$ is reachable from $e_{\mathbf{x}_i}^{\mathrm{out}}$ by a local path and an adjacency exists from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$ in $\mathrm{P}(G, F)$, using an empty list of nesting patterns as the adjacency certificate.

Otherwise, $j > i + 1$ and there are $j - i$ nested patterns between $\mathbf{x}_i$ and $\mathbf{x}_j$. Rename the nesting patterns between $\mathbf{x}_i$ and $\mathbf{x}_j$ as $\mathbf{z}_1, \ldots, \mathbf{z}_{j-i}$ where $\mathbf{z}_{i'} = \mathbf{x}_{i+i'}$. If $\mathbf{z}_1, \ldots, \mathbf{z}_{j-i}$ compose an adjacency certificate for $\mathbf{x}_i \to \mathbf{x}_j$, then this edge exists in $\mathrm{P}(G, F)$. Otherwise, there exists such a k that violates the adjacency condition between $\mathbf{x}_i$ and $\mathbf{x}_j$, then let i' be the smallest such index. There is an edge in $\mathrm{P}(G, F)$ from $\mathbf{x}_i$ to the nesting pattern description $\mathbf{z}_{i'}$, since $\mathrm{Tail}(e_{\mathbf{z}_{i'}}^{\mathrm{in}})$ is reachable from $\mathrm{Head}(e_{\mathbf{x}_i}^{\mathrm{out}})$ by a path using the nesting patterns $\mathbf{z}_1, \ldots, \mathbf{z}_{i'-1}$ as the adjacency certificate. By Lemma 37, $\mathrm{Tail}(e_{\mathbf{x}_j}^{\mathrm{in}})$ is reachable from $\mathrm{Head}(e_{\mathbf{z}_{i'}}^{\mathrm{out}})$ using an irreducible path which induces the patterns $\mathbf{z}_{i'+1}, \ldots, \mathbf{z}_{j-i}$. By iteration, there is a path from $\mathbf{z}_{i'}$ to $\mathbf{x}_j$ in $\mathrm{P}(G, F)$, and hence a path from $\mathbf{x}_i$ to $\mathbf{x}_j$ in $\mathrm{P}(G, F)$. Connecting all of the edges between the full patterns in $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ gives a path from u' to v' in $\mathrm{P}(G, F)$.

($\Leftarrow$) Given a path $P = u', \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\ell, v'$ in $\mathrm{P}(G, F)$, let $\mathbf{x}_j = (i_j, t_j, o_j, p_j)$ for each $j \in \{1, \ldots, \ell\}$. Since $u' \to \mathbf{x}_1$ in $P(G)$, E_{i_1} is a class incident to T_u and all edges are reachable from u. Specifically, there is a tree path P_0 from u to $e_{\mathbf{x}_1}^{\mathrm{out}}$. Similarly, since $\mathbf{x}_\ell \to v'$ in $\mathrm{P}(G, F)$, E_{i_k} is a class incident to T_v and all edges have v as a head. For each $j \in \{1, \ldots, \ell - 1\}$, Lemmas 31 and 37 imply there is an irreducible path P_i in G from the head of $e_{\mathbf{x}_j}^{\mathrm{out}}$ to the tail of $e_{\mathbf{x}_{j+1}}^{\mathrm{in}}$ that is either a local path or induces a list of nesting pattern descriptions which form an adjacency certificate. Also, by Definition 30, there exist (possibly empty) paths Q_j from $e_{\mathbf{x}_j}^{\mathrm{in}}$ to $e_{\mathbf{x}_j}^{\mathrm{out}}$ using local paths and edges of the class E_{i_j}. These paths concatenate to a path $uP_0e_{\mathbf{x}_1}^{\mathrm{out}}P_1e_{\mathbf{x}_2}^{\mathrm{in}}Q_2e_{\mathbf{x}_2}^{\mathrm{out}}P_2e_{\mathbf{x}_3}^{\mathrm{in}} \ldots e_{\mathbf{x}_{\ell-1}}^{\mathrm{out}}P_{\ell-1}e_{\mathbf{x}_\ell}^{\mathrm{in}}v$ from u to v in G. ∎

Lemma 40. *The pattern graph $\mathrm{P}(G, F)$ is log-space computable.*

Proof: Given a pattern description $\mathbf{x}$, we describe a log-space algorithm for enumerating the pattern descriptions reachable by an edge in $\mathrm{P}(G, F)$. It is simple to find the pattern descriptions $\mathbf{x}, \mathbf{y}$ so that $u \to \mathbf{x}$ and $\mathbf{y} \to v$.

A necessary subroutine takes a global edge e and enumerates all pattern descriptions reachable from $\mathrm{Head}(e)$ using local paths in the exit direction of $\mathbf{x}$. By Lemma 27, there is an ordered list of topological equivalence classes $E_{i_0}, E_{i_1}, \ldots, E_{i_\ell}$ reachable by local paths from the head of e. E_{i_0} is the class containing e, so e is in $\mathcal{R}[E_{i_0}]$. All other classes E_{i_j} (for $j \geq 1$, except possibly $j = \ell$) are fully reachable. Hence, each pattern description $\mathbf{y}$ centered at a class E_{i_j} with $j \in \{1, \ldots, \ell - 1\}$ (where the entrance direction of $\mathbf{y}$, orientation, and end all match the exit direction of $\mathbf{x}$) has $e_{\mathbf{y}}^{\mathrm{in}}$ reachable from $\mathrm{Head}(e)$ using a local path. Each pattern description $\mathbf{y}$ with entering direction the same as the exit direction of $\mathbf{x}$ and centered at E_{i_ℓ} can be

checked if $e^{\text{in}}_{\mathbf{y}}$ is reachable from e. The only pattern that could be used without having $e^{\text{in}}_{\mathbf{y}}$ reachable is a nesting pattern.

To enumerate all neighbors of $\mathbf{x}$ in $\mathrm{P}(G, F)$, perform the above subroutine on $e^{\text{out}}_{\mathbf{x}}$, adding edges from $\mathbf{x}$ to each reachable pattern description $\mathbf{y}$. If the nesting pattern $\mathbf{z}$ on E_{i_ℓ} is not fully reachable (i.e. there is no local path from e to $e^{\text{in}}_{\mathbf{z}}$ in the proper direction) then compute the most-interior edge $e^{\text{int}(\text{Head}(e))}_{\mathbf{z}}$. Repeat the subroutine on this edge, continuing until the class E_{i_ℓ} is fully reachable (or the list is empty). In the jth iteration, let $w_{j-1} = \text{Head}(e)$ and $\mathbf{z}_j = \mathbf{z}$.

It is clear this algorithm takes log-space. It enumerates all neighbors of $\mathbf{x}$ in $\mathrm{P}(G, F)$, since a neighbor $\mathbf{y}$ requires a list of nesting classes $\mathbf{z}_1, \ldots, \mathbf{z}_\ell$ so that there is an irreducible path from $\mathbf{x}$ to $\mathbf{y}$ inducing these classes. Each class $\mathbf{z}_j$ has the edge $e^{\text{in}}_{\mathbf{z}_j}$ *not* reachable from $\mathbf{x}$ using the patterns $\mathbf{z}_1, \ldots, \mathbf{z}_{j-1}$. This means that the pattern $\mathbf{z}_j$ is centered at the class E_{i_ℓ} computed by the iteration of the subroutine on the edge $e^{\text{int}(w_{j-1})}_{\mathbf{z}_{j-1}}$. Moreover, $\mathbf{y}$ appears as a reachable class from the most-interior edge computed at $\mathbf{z}_\ell$, so $\mathbf{y}$ is enumerated. Finally, any pattern enumerated by this procedure can reconstruct the list of $\mathbf{z}_1, \ldots, \mathbf{z}_\ell$ by using the nesting patterns used in the subroutine iterations. ■

The main theorem is found by combining Lemma 26, Theorem 39, and Lemma 40, using $G' = P(G, F)$ where the forest decomposition F is given by the lexicographically least incoming edge.

ACKNOWLEDGEMENTS

We thank Jeff Erickson for sharing his knowledge on topological embeddings of graphs. We also thank Jonathan F. Buss for discussions on simultaneous time-space bounds for reachability at the 2010 Conference on Computational Complexity. Thanks to the anonymous referees whose comments improved this paper.

REFERENCES

[1] O. Reingold, "Undirected connectivity in log-space," *Journal of the ACM*, vol. 55, no. 4, 2008.

[2] O. Reingold, L. Trevisan, and S. Vadhan, "Pseudorandom walks on regular digraphs and the RL vs. L problem," in *STOC '06: Proceedings of the thirty-eighth annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM, 2006, pp. 457–466.

[3] A. Wigderson, "The complexity of graph connectivity," *Mathematical Foundations of Computer Science 1992*, pp. 112–132, 1992.

[4] E. Allender, "Reachability problems: An update," *Computation and Logic in the Real World*, pp. 25–27, 2007.

[5] W. J. Savitch, "Relationships between nondeterministic and deterministic tape complexities," *Journal of Computer and System Sciences*, vol. 4, no. 2, pp. 177–192, 1970.

[6] M. Saks and S. Zhou, "$\text{BP}_H\text{SPACE}(S) \subseteq \text{DSPACE}(S^{3/2})$," *Journal of Computer and System Sciences*, vol. 58, no. 2, pp. 376–403, 1999.

[7] A. Jakoby, M. Liśkiewicz, and R. Reischuk, "Space efficient algorithms for directed series-parallel graphs," *Journal of Algorithms*, vol. 60, no. 2, pp. 85–114, 2006.

[8] A. Jakoby and T. Tantau, "Logspace algorithms for computing shortest and longest paths in series-parallel graphs," in *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science*, 2007, pp. 216–227.

[9] E. Allender, D. A. M. Barrington, T. Chakraborty, S. Datta, and S. Roy, "Planar and grid graph reachability problems," *Theory of Computing Systems*, vol. 45, no. 4, pp. 675–723, 2009.

[10] D. Stolee, C. Bourke, and N. V. Vinodchandran, "A log-space algorithm for reachability in planar acyclic digraphs with few sources," *25th Annual IEEE Conference on Computational Complexity*, pp. 131–138, 2010.

[11] N. Nisan, "RL $\subseteq$ SC," in *In Proceedings of the Twenty Fourth Annual ACM Symposium on Theory of Computing*, 1995, pp. 619–623.

[12] G. Barnes, J. F. Buss, W. L. Ruzzo, and B. Schieber, "A sublinear space, polynomial time algorithm for directed s-t connectivity," in *Structure in Complexity Theory Conference, 1992., Proceedings of the Seventh Annual*, 1992, pp. 27–33.

[13] C. Thomassen, "The graph genus problem is NP-complete," *Journal of Algorithms*, vol. 10, no. 4, pp. 568–576, 1989.

[14] J. Chen, S. Kanchi, and A. Kanevsky, "A note on approximating graph genus," *Information processing letters*, vol. 61, no. 6, pp. 317–322, 1997.

[15] J. Kynčl and T. Vyskočil, "Logspace reduction of directed reachability for bounded genus graphs to the planar case," *ACM Transactions on Computation Theory*, vol. 1, no. 3, pp. 1–11, 2010.